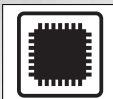
Design Wave 設計コンテスト 2008
「RSA暗号化器の設計」

デバイスの記事

ビギナーズ

吉田たけお，尾知 博

今回の設計課題は，RSA暗号の暗号化器と復号器の設計である．琉球大学の学内で開催した第２回（1999年）の設計課題と同じテーマだが，オープンなコンテストである今回の課題とする．（筆者）

設計課題はRSA暗号です．まず，暗号に関する基本的な用語について説明します．

1 暗号に関する基礎知識

暗号とは，発信者（Sender）が，特定の受信者（Receiver）以外には知られたくない情報やデータを，その特定の受信者以外には分からないように秘匿，あるいは偽装する手法のことをいいます．

● 平文による通信

発信者が特定の受信者に伝えたい情報やデータのことを平文（Plaintext）といいます．平文は，誰にでもその意味や内容を把握できる表現になっています．

発信者は，この平文を，電話回線や郵便，コンピュータ・ネットワークなどの通信路（Communication Channel）を介して，受信者に伝送します．しかし，これらの通信路は，第三者に盗聴（Tapping）される危険性があります．このとき，情報を不正に入手しようとする第三者のことを盗聴者（Wiretapper）といいます．

何の対策も施していない場合，盗聴者に，平文をそのまま入手され，さらに，発信者が受信者に伝えたい情報を把握されてしまいます．この様子を**図1**に示します．

● 暗号を用いた通信

暗号は，このような情報のやりとりにおいて，盗聴者にその内容が分からないようにするために用いられます．

暗号を用いた通信では，平文の意味や内容を盗聴者が把握できない（把握しにくい）ように，平文を変換してから，受信者に送ります．この平文を変換したものを暗号文（Ciphertext）といいます．

ここで，平文を暗号文に変換することを暗号化

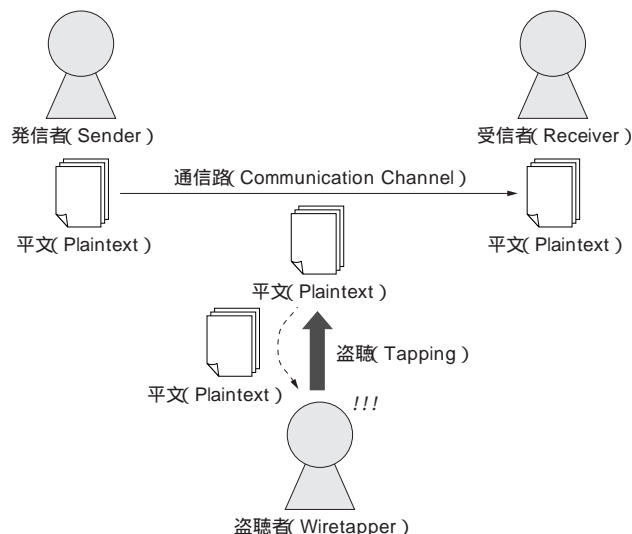



図1　暗号を用いない通信

KeyWord　RSA暗号，平文，盗聴，暗号文，暗号化，復号化，公開鍵，文字コード，べき乗，mod，割り算

（Encryption），暗号化を行うディジタル回路を暗号化器（Encipher）といいます．また受信者が，暗号文を元の平文に戻すことを復号化（Decryption）あるいは単に復号といい，復号を行うディジタル回路を復号器（Decipher）といいます．盗聴者が不正に入手した暗号文を元の平文に戻すことは解読といいます．

暗号文は，受信者には簡単に復号できて，盗聴者には簡単に解読できないようになっている必要があります．そこで発信者は，パスワードを知っている者だけが簡単に復号できるように，暗号文を作ります．このパスワードのことを，暗号の世界では，鍵（Key）といいます．発信者は事前に，受信者だけに鍵を渡しておき，その後，暗号文を送ります．受信者は，受け取った暗号文を，発信者に渡された鍵を使って復号し，元の平文を得ます．一方，盗聴者は，不正に暗号文を入手できますが，鍵がないので，簡単には解読できません．

この様子を**図2**に示します．

図2　暗号を用いた通信

● 秘密鍵暗号と公開鍵暗号

暗号には，さまざまの種類が存在しますが，それらは秘密鍵暗号と公開鍵暗号に大別できます．今回の設計課題であるRSA暗号は，公開鍵暗号の一種です．

暗号における鍵は，暗号化と復号の両方の処理において用いられます．暗号化の際に用いられる鍵を暗号化鍵（Encryption Key），復号の際に用いられる鍵を復号鍵（Decryption Key）といいます．ここで，暗号化鍵と復号鍵が同じであるような暗号のことを秘密鍵暗号，暗号化鍵と復号鍵が異なるような暗号のことを公開鍵暗号といいます．

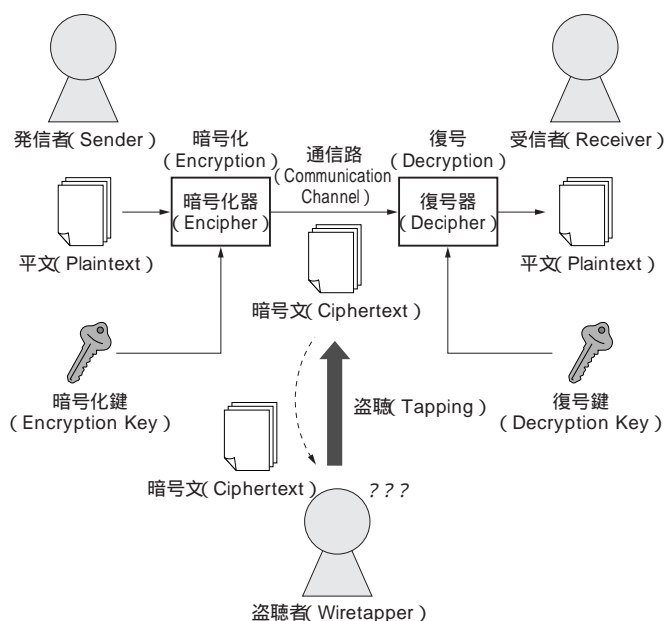
秘密鍵暗号では，暗号化鍵と復号鍵が同じなので，両者を区別せずに，単に鍵と呼びます．また通常，この鍵は，発信者が作成（生成）し，安全な方法を使って受信者に渡します．秘密鍵暗号を使った通信において，安全性を確保するためには，この「安全な方法で鍵を渡す」こと，すなわち，鍵の情報を第三者に漏らさないこと，が重要になります．このように，鍵を秘密にしておく必要があるので，秘密鍵暗号と呼ばれています．

一方，公開鍵暗号では，暗号化鍵と復号鍵が異なります．このことを，ドアの鍵に例えると，施錠用の鍵と開錠用の鍵が異なることを意味します．自宅のドアがこのような構造になっていると，不便で仕方ありません．しかし暗号においては，このしくみは絶大な威力を発揮します．公開鍵暗号の暗号化鍵と復号鍵は，受信者が作成します．そして，暗号化鍵を世間に公開し，復号鍵は秘密にしておきます．このため，暗号化鍵のことを公開鍵（Public Key），復号鍵のことを秘密鍵（Private Key）とも呼びます．

● 公開鍵暗号による通信のしくみ

暗号化鍵を世間に公開するということは，盗聴者にも公開するという意味です．ここで，公開鍵暗号を使った通信の概要を，**図3**を用いて説明します．

いま，AさんとBさんの間で，公開鍵暗号を使った通信を行うものとします．Aさんは，独自に，暗号化鍵（公開鍵）E_Aと復号鍵（秘密鍵）D_Aを作り，このうち，公開鍵E_Aを公開します．同様にBさんも，独自に，暗号化鍵（公開鍵）E_Bと復号鍵（秘密鍵）D_Bを作り，このうち，公開鍵E_Bを公開します．

AさんがBさんあてに平文P_Bを送る場合は，まずBさんによって公開されている公開鍵E_Bを入手します．Aさんは，この公開鍵E_Bを使って平文P_Bを暗号化し，暗号文C_Bを作り，これをBさんに送ります．Bさんは，秘密鍵D_Bを使って暗号文C_Bを復号し，Aさんからの平文P_Bを入手します．このとき，盗聴者が入手可能な情報は，Bさんによって公開されている公開鍵E_Bと，通信路を盗聴して得られる暗号文C_Bです．Bさんが，秘密鍵D_Bを秘密にしておけば，暗号文C_Bを復号できるのはBさんだけなので，盗

聴者に平文P_Bの内容が漏れることはありません．なお，暗号文C_Bを作ったAさんも，秘密鍵D_Bを知らないので，暗号文C_Bを元の平文P_Bに復号できません（平文P_Bを作ったのはAさんなので，復号する必要はない）．この点が，公開鍵暗号の特徴でもあります．

Bさんが A さんあてに平文P_Aを送る場合も同様です．まず，A さんによって公開されている公開鍵E_Aを入手します．Bさんは，この公開鍵E_Aを使って平文P_Aを暗号化し，暗号文C_Aを作り，これを A さんに送ります．この場合も，A さんが，秘密鍵D_Aを秘密にしておけば，盗聴者が入手可能な情報から平文P_Aの内容が漏れることはありません．

このように，暗号化鍵を世間に公開するので，公開鍵暗号と呼ばれています．

図3の例では，二人の間の通信を例にしましたが，A さんと B さんが公開している暗号化鍵を用いれば，誰でも，A さんやBさんあての暗号文を作成できます．公開鍵暗号は，秘密鍵暗号で必要な「安全な方法で鍵を渡す」ことが不要となっています．このことを可能にしているのは，暗号化鍵と復号鍵を区別している点にあります．

公開鍵暗号は，画期的な暗号としてさまざまな分野で使われています．しかし，暗号化および復号に要する処理時間が秘密鍵暗号と比較して，非常に長いという短所があります．そのため実際には，両者を組み合わせて使用します．すなわち，通常の暗号通信には，高速処理が可能な秘密鍵暗号を用いますが，その秘密鍵暗号の鍵を公開鍵暗号を使ってやりとりします．こうすれば，両者の短所を補って，安全な暗号通信が行えます．

2 RSA暗号について

公開鍵暗号では，公開している暗号化鍵から，秘密にしている復号鍵を容易には求められないしかけが必要になります．このしかけを初めて実現した，世界初の公開鍵暗号がRSA 暗号です．

RSA 暗号は，米国 MIT（Massachusetts Institute of Technology）に勤務していた Rivest，Shamir，Adleman の3名の研究者によって開発されたので，この3名の頭文字をとってRSA と命名されました．

RSA 暗号は，コンピュータを使う人であれば，誰でも，知らないうちに使っている可能性があります．そのくらい，現在までに広く実用化されています．具体的には，電子メールやファイルの暗号化に用いられるPGP（Pretty Good Privacy）や電子商取引などに用いられる SSL（Secure Socket Layer）などがあります．また詳細は割愛しますが，RSA 暗号は，電子署名も実現できます．

このように，広く普及しているRSA 暗号ですが，その暗号化と復号の処理では，非常に大きな値を扱うため，処理時間が長くなるといった問題点もあります．そのため，暗号化や復号を行うハードウェア製品も数多く発売されています．

● 暗号化と復号の手順

今回の課題は，RSA 暗号の暗号化と復号を行う回路を実現することです．このRSA 暗号を一言で説明すると，二つの素数の積を求める手間に比べて，その積を素因数分解する手間の方がとても難しく時間がかかるという性質を利用した暗号，ということになります．何やら難しそうな印象を受けるかもしれませんが，RSA 暗号の暗号化と復号の

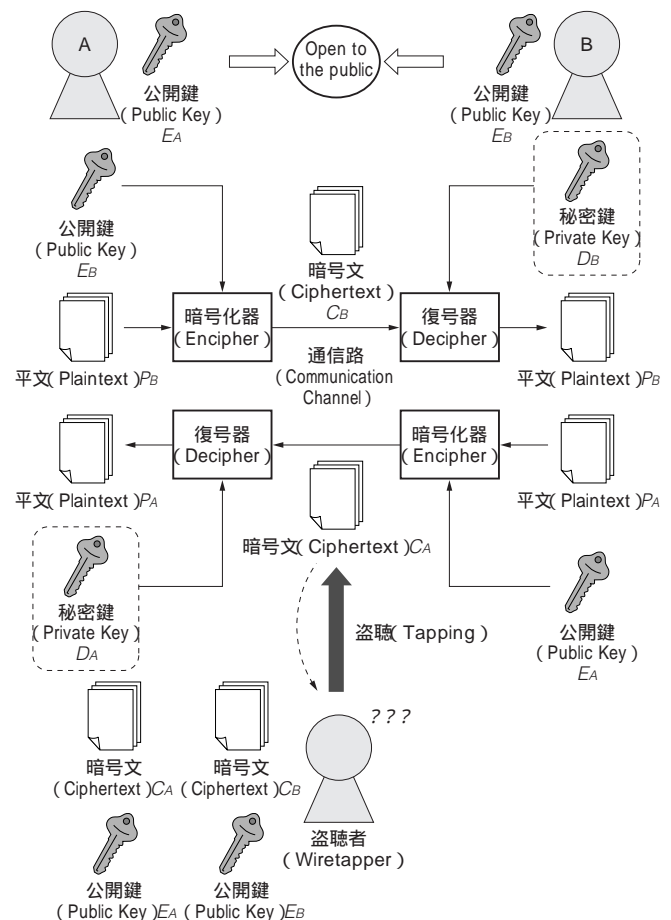



図3　公開鍵暗号を用いた通信

手順は，非常に単純です．

今，平文P，暗号文C，暗号化鍵E，復号鍵D，法Mのそれぞれが，正整数で表されているものとします．このとき，RSA暗号では，以下の式を用いて，暗号化を行います．

$$C = P^E \bmod M \quad \cdots\cdots(1)$$

ここで，modは剰余を求める演算子で，$A \bmod B$は，AをBで割った余りを表します．またRSA暗号では，以下の式を用いて復号を行います．

$$P = C^D \bmod M \quad \cdots\cdots(2)$$

● 暗号化鍵と復号鍵の作り方

暗号化鍵E，復号鍵D，法Mの作り方を説明します．

まず，非常に大きな二つの素数p，$q\ (p \neq q)$をランダムに選びます．このとき，法Mは，

$$M = p \times q$$

とします．次にp，qから，

$$L = \mathrm{LCM}(p-1, q-1) \quad \cdots\cdots(3)$$

を求めます．ここで，LCM$(a,\ b)$は，aとbの最小公倍数を表します．このとき，式(3)のLと互いに素，かつ，L未満となる数を選び，その数を暗号化鍵Eとします．すなわち，暗号化鍵Eは，

$$\mathrm{GCD}(L, E) = 1 \quad \cdots\cdots(4)$$

を満たす，L未満の数とします．ここで，GCD$(a,\ b)$は，aとbの最大公約数を表します．なお，暗号化の処理時間を短く済ませるために，暗号化鍵Eには，小さな値がよく用いられます．

復号鍵Dは，ある正整数Hに対して，以下の式を満たす数とします．

$$E \cdot D = H \cdot L + 1 \quad \cdots\cdots(5)$$

以上の方法で得られる暗号化鍵E，復号鍵D，法Mを用いれば，先に示した式(1)と式(2)を用いて，暗号化と復号を行えます．RSA暗号では，法Mと暗号化鍵Eを公開しておき，復号鍵Dを秘密に保持しておきます．

● けた数の多い鍵と法を用いれば解読は困難になる

RSA暗号では，素因数分解の難しさを利用して，第三者が暗号文を解読することを困難にしています．鍵や法の作り方からも分かると思いますが，RSA暗号では，法Mを素因数分解できれば，復号鍵Dを簡単に求めることができてしまいます．

しかし，法Mが数百けた以上になる場合，その素因数分解に要する時間は，最新のコンピュータを使っても数年から数十年以上を要します．すなわち，RSA暗号を実際に使用する場合，できるだけけた数の多い鍵と法を用いる必要があります．

3 RSAによる暗号化と復号の例

RSA暗号の暗号化と復号の具体例を示します．

● 暗号化鍵，復号鍵，法の決定

まず，暗号化鍵E，復号鍵D，法Mを決めておく必要があります．そこで二つの素数p，$q\ (p \neq q)$を選びます．ここでは，それぞれ2けた程度の数とし，$p = 11$，$q = 23$とします．これらの数から法Mが求まります．

$$M = p \times q = 253$$

また，

$$L = \mathrm{LCM}(11-1,\ 23-1) = 110$$

となるので，暗号化鍵Eとして101を選ぶことにします．実際に，GCD(110,　101)＝1となることを確認してみてください．

次に，これらの数を式(5)に代入し，$H = 1, 2, 3, \cdots$について，整数になるDを探すと，$H = 56$のとき，$D = 61$となることが分かります．

● 文字コードとブロック・サイズの決定

暗号化鍵E，復号鍵D，法Mのほかに，平文の表現に用いる文字コードとブロック・サイズの値を決めておく必要があります．

ブロック・サイズとは，一度に暗号化を行う文字数のことです．通常，平文の文字数は膨大になります．そのため，平文の内容をいくつかのブロックに分け，個々のブロックについて式(1)を用いて暗号化を行います．このときの一

つのブロックの文字数がブロック・サイズになります．

実際の暗号化ではこのブロック・サイズも大きな値にする必要があります．今回は簡単のために，1文字ずつ暗号化するものとします．また，平文に使用する文字コードですが，今回は，ASCII コード（**表1**）を使用します．

● 暗号化

RSA 暗号の暗号化と復号の例を見てみましょう．まず，平文の例を示します．

Enjoy HDL!

表1 の ASCII コード表から，上記の平文を文字コードで表現すると，以下のようになります．

69 110 106 111 121 32 72 68 76 33

平文の最初の文字“E”を ASCII コードで表すと 69 になります．これを式(1)に代入すれば，暗号化された文字の文字コードが得られます．すなわち，

$$C = P^E \bmod M = 69^{101} \bmod 253 \quad \cdots\cdots(6)$$

を計算すればよいわけです．しかし，P^E の結果は，けた数が非常に多くなるため，筆算や普通の電卓では計算できないと思います．幸い，この程度のけた数であれば，ちょっとしたプログラムを作ったり，けた数の多い数を扱える電卓ソフトなどを使えば，比較的簡単に計算結果を確認できます．これらの方法で確認した結果，式(6)の結果はたまたま同じ 69 となりますが，以下のような暗号文が得られました．

69 209 172 122 220 219 193 68 43 176

● 復号

この暗号文を平文に戻すには，式(2)を使います．例えば，暗号文の最初の文字 69 を復号するには，

$$P = C^D \bmod M = 69^{61} \bmod 253 \quad \cdots\cdots(7)$$

を計算すればよいわけです．式(7)の結果は 69 になります．

残りの文字についても，暗号化の計算と同じようにして確認したところ，元の平文の文字コードが得られます．

4 RSA 暗号化器のハードウェア実装に関する注意点

設計課題のシステム概要を**図4**に示します．課題は，RSA 暗号の暗号化器と復号器の設計です．とはいっても，式(1)と式(2)を比べると分かると思いますが，RSA 暗号の暗号化器と復号器は同じになります．ですから実際は，暗号化器（復号器）のみを設計すればよいことになります．

ここでは，式(1)を実現する回路を設計する際の注意点を述べておきます．式(1)は，非常に単純な式ですが，式(1)で用いられている 2 種類の演算，すなわち，べき乗の計算と mod 演算は，どちらも論理合成できない演算です．今回の課題では，これらの演算をどのように実現するのかがポイントになります．以下では，これらの演算を，論理合成可能な演算子だけを用いて実現する方法を示します．

● べき乗の計算の工夫

式(1)を言葉で説明すると，「暗号文 C は，平文 P を E 乗し，その結果を M で割った余り」となります．すなわち，

表1　ASCII 文字コード表

10進数	文字	10進数	文字	10進数	文字	10進数	文字
0	NUL	32	SP	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	EXT	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

※0～31 と 127 は制御文字．32 の SP はスペース（空白文字）．

$C < M$が成り立ちます．もちろん，同様に，$P < M$も成り立ちます．しかし，途中結果である「平文PをE乗」した結果は，非常に大きな値になります．

先ほどの暗号化の例に登場した69^{101}は，10進数で200けた近い値になります．たった1文字を暗号化するために，このように大きな値を回路内部に保持することは，現実的ではありません．ですから，この計算には工夫を要します．

幸い，mod演算では，

$$(x \times y) \bmod p = ((x \bmod p) \times (y \bmod p)) \bmod p \quad \cdots\cdots(8)$$

という式が成り立ちます．すなわち，複数の数を掛けてから，最後にmod演算をする場合，途中でmod演算を施しても，計算結果は同じになります．この式を用いて，例えば，式(9)のように，乗算を行うたびに剰余を計算しておけば，途中の計算結果が非常に大きくなることを防げます．

$$\begin{aligned} C &= P^E \bmod M \\ &= (\cdots((P \bmod M) \times (P \bmod M) \bmod M) \times \cdots) \bmod M \end{aligned} \quad \cdots(9)$$

● べき乗の計算のさらなる工夫

この例では，計算の途中結果がM^2を超えることはありません．しかし，この方法では，E回の乗算が必要になります．先ほどの暗号化の例でさえ，101回もの乗算が必要になってしまいます．安全性を考慮した実用的なサイズの暗号化鍵Eを用いる場合は，101とは比較できないほど大きな数を用いるので，さらに非現実的になってしまいます．

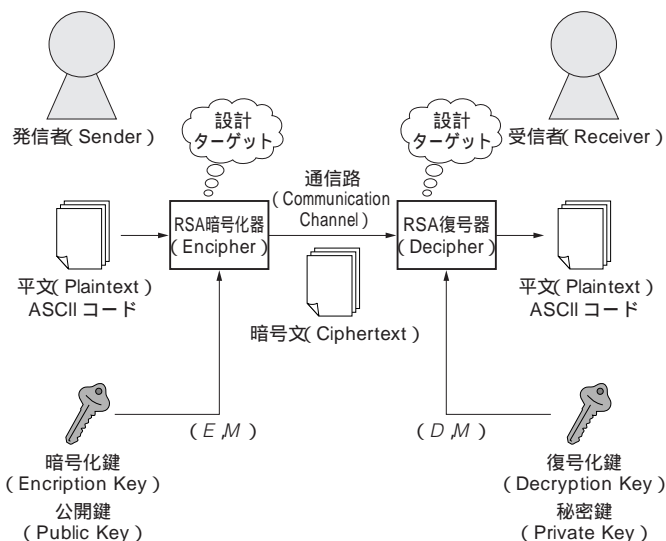



図4　設計課題のシステム概要

ですから，もうひと工夫する必要があります．ここでは，よく用いられる工夫を説明します．

まず，10進数表現の暗号化鍵Eを2進数を使って，

$$\begin{aligned} &E = (a_k a_{k-1} \cdots a_1 a_0)_2 \\ &\text{ここで } a_i \in \{0,1\}, i = 0,1,\cdots,k \end{aligned} \quad \cdots\cdots(10)$$

のように表します．このとき，

$$\begin{aligned} E &= a_k \times 2^k + a_{k-1} \times 2^{k-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0 \\ &= \sum_{i=0}^{k} a_i 2^i \end{aligned} \quad \cdots(11)$$

となります．この式を用いると，

$$\begin{aligned} P^E &= P^{(a_k a_{k-1} \cdots a_1 a_0)_2} = P^{\sum_{i=0}^{k} a_i 2^i} \\ &= P^{a_k 2^k} \times P^{a_{k-1} 2^{k-1}} \times \cdots \times P^{a_1 2^1} \times P^{a_0 2^0} \end{aligned} \quad \cdots\cdots(12)$$

と表せます．さらに，式(9)と式(12)を用いると，

$$\begin{aligned} P^E \bmod M = (\cdots((P^{a_0 2^0} \bmod M) \\ \times (P^{a_1 2^1} \bmod M) \bmod M \cdots)) \bmod M \end{aligned} \quad \cdots(13)$$

となります．つまり，P^{2^0}の計算，P^{2^1}の計算，…，P^{2^k}の計算と続けていき，必要に応じて，それらを掛け合わせればよいことになります．ここで，「必要に応じて」とは，

$$P^{a_i 2^i} = \begin{cases} 1 & a_i = 0 \text{ のとき} \\ P^{2^i} & a_i = 1 \text{ のとき} \end{cases} \quad (i = 0,1,\cdots,k) \quad \cdots\cdots(14)$$

となるので，「$a_i = 1$のときだけ掛ければよい」という意味です．また，式(13)において，乗算のたびにmod演算を行っておけば，途中の計算結果はM^2を超えることはありません．

ここで，肝心の，$P^{a_i 2^i} \bmod M (i = 0,1,\cdots,k)$は，

$$P^{a_i 2^i} \bmod M = \left(\cdots\left(\left(P^2 \bmod M\right)^2 \bmod M\right)^2 \bmod M \cdots\right)^2 \bmod M \quad \cdots(15)$$

のように，i回の2乗とmod演算で計算できます．

数式がたくさん登場しましたが，結局のところ，式(13)と式(15)を用いれば，式(1)を計算できることになります．このとき，途中結果の値は，M^2を超えません．また，乗算やmod演算の回数も，Eの値そのものではなく，そのけた数(Eのlogオーダ)に比例した回数で済みます．

図5に，式(13)と式(15)を用いた場合の暗号化の流れを示します．**図5**を見ると，規則的な流れになっていることが分かると思います．以下，この流れについて説明します．

まず，**図5**の例では，暗号化鍵の長さを4ビットとし，

$$E=(a_3,\ a_2,\ a_1,\ a_0)^2$$

と表しています．鍵の長さが4ビットなので，最大P^{15}を計算する必要があります．**図5**の上側では，最初に，$P\times P=P^2$を求め，次に，$P^2\times P^2=P^4$を求め，最後に，$P^4\times P^4=P^8$を求めていることが分かります．

すなわち，式(15)に従って，P^2，P^4，P^8を次々に計算しています．もちろん，乗算のたびに，mod演算をしていますので，途中の計算結果はM^2を超えません．

一方**図5**の下側では，最初に，$1\times 1(=1)$，$P\times 1(=P)$のいずれかの計算を行っています．どちらの計算を行うのかは，式(14)に従って，a_0の値で判断します．実際には，この乗算は1を掛けるだけの無意味な計算なので，**図5**では乗算とmod演算を省略していますが，便宜上，このように説明しています．次に，この結果に対して，1またはP^2を掛けています．どちらを掛けるかは，同じく式(14)に従ってa_1の値で判断します．この結果は，a_0とa_1の値の組み合わせによって，1，P，P^2，P^3のいずれかになります．以下同様に，1またはP^4を掛け，その結果に，1またはP^8の結果を掛けます．このような手順で計算すると，$E=(a_3,\ a_2,\ a_1,\ a_0)$の値に応じて，$P^0(=1)$，$P^1\ (=P)$，$P^2$，$P^3$，…，$P^{15}$のいずれかの値が計算され，mod演算の後，暗号文$C$として出力されます．

なお，暗号化鍵の長さが長くなった場合でも，乗算とmod演算の回数が増えるだけで，すなわち，**図5**が右方向に伸びるだけで，基本的な構造は同じままで暗号化ができます．

● modの計算

mod演算のアルゴリズムやハードウェア化の方法には，さまざまな方法があります．また，加算の後のmod演算や乗算の後のmod演算など，特定のケースでmod演算を行う場合，それに特化して高速化を行う方法もあります．ここでは，凝った方法の紹介は割愛し，単純な方法を一つだけ紹介します．

mod演算は，剰余(割ったあまり)を求める演算なので，結局は，割り算(除算)を行うことで求められます．ですから，結果がB未満になるまで，AからBを引き続ければ，A mod Bを計算できます．しかし，この方法は，式(9)でべき乗を計算するのと同様に，非常に効率が悪くなるので，やはり工夫が必要になります．

ここで登場するのが，小学校で習った筆算による割り算です．筆算による割り算とは，紙に数字を書いて，割り算を計算する方法です．例えば，12345÷171を計算する場合，**図6**に示すように，まず「？」の部分がいくつになるのかの見当をつけます．試しに，「？」を8として計算すると，余りが－134となってしまい，仮定した値が大きすぎることが分かります．そこで「？」を7として計算すると，今度はうまくいくので，次のけたの見当に移ります．このように，筆算の割り算では，商の値を仮定し，それに除数を掛けた値を，被除数から引く，という手順を繰り返します．なお，**図6**の計算を最後まで行うと，**図7**に示すように，12345÷171の結果は，商が72，あまりが33となります．

さて，ディジタル回路の内部では，通常，2進数が扱われます．今回の課題では，この2進数同士の割り算をする回路を設計する必要があります．この割り算回路を実現す

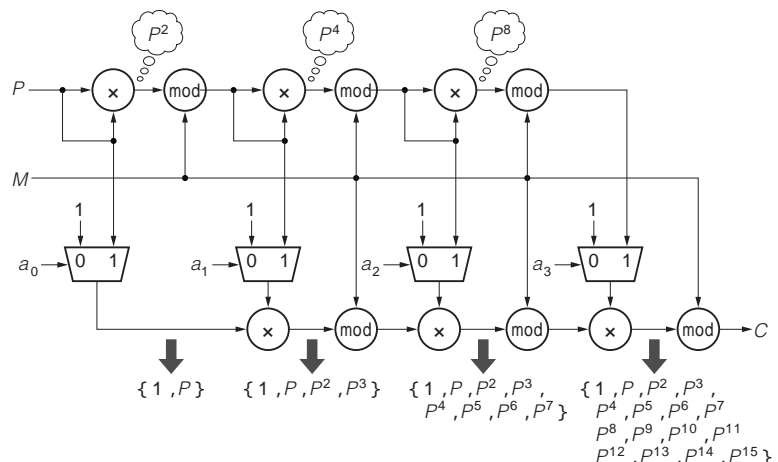



図5
暗号化手順(Eが4ビットの場合)
本文の式(13)と式(15)を用いた場合を示す．

る上で，筆算による割り算の手順が役に立ちます．

図6に示したように，10進数の割り算の場合，仮定する商は，0～9までの10通りあります．しかし，2進数での割り算の場合，0または1の2通りしかありません．すなわち，引けるか，引けないか，だけを考えればよくなり，10進数の割り算よりも単純になります．例えば，2進数の割り算11011110÷1010を，筆算で行うと，**図8**のようになります．

図8で行われている処理は，

- 引けるか，引けないかを判断するための大小比較
- 次のけたに移る処理
- 引き算

の三つです．これらの処理を，けた数に比例した回数繰り返すことによって，割り算を行えます．なお，これら三つの処理は，いずれも論理合成できます．

● 演算性能の改善への考え方

以上で説明した計算方法は，あくまでも一例で，改善の余地がたくさん残されています．そこで，残された改善の余地について補足しておきます．

べき乗は**図5**に従って計算することが可能です．例えば，**図5**をそのままハードウェア化することも可能です．この場合，使用される構成要素は，

- 乗算器
- セレクタ（マルチプレクサ）
- mod演算器

の三つです．mod演算器に関しては後で説明しますが，乗算器とセレクタは，HDLで簡単に記述でき，論理合成も可能です．ここで注意する必要があるのは，乗算器です．VHDLでも，Verilog HDLでも，乗算の記号は「*」です．この記号「*」は，合成可能ですが，非常に大きな回路が合成されます．今回の課題であるRSA暗号化器では，通常，1000ビットとか2000ビットといった非常に大きな数を扱いますが，このように大きな数の乗算にそのまま「*」を用いても，コンピュータの処理能力上，まず合成できないでしょう．仮に合成できたとしても，その回路は使い物にはなりません．今回の課題では，1000ビットとか2000ビットといった大きな数は扱わないので，「*」の使用も可能です．課題に取り組む際には，この点を念頭に置き，余力があれば改善してください．

これと同じようなことが，mod演算器の大小比較でもいえます．先ほど説明した手順をmod演算器に適用する場合，引けるか，引けないかを判断するための大小比較器が必要になります．HDLでは，関係演算子を使って大小を比較でき，合成可能です．しかし，大きな数同士の大小比較は，効率が良いとはいえません．除算器について調べてもらえれば分かると思いますが，実際には大小の比較を行わず，いきなり引き算をします．この先の対処方法，すなわち，引き算の結果が負であった場合の対処方法には，2種類あります．一つは，引いた値を足して，元に戻してから，次のけたに移る方法です．もう一つは，そのまま次のけたに移り，そのけたでは引かずに足す，という方法です．それぞれ，引き戻し法，引き放し法と呼ばれています．ここではこれ以上触れませんが，mod演算器を設計する際の参考にしてください．もちろん，今回の課題で扱う数はそれほど大きくないので，関係演算子を使用することも可能です．しかし，余力があれば，この点についても改善を試みてください．

RSA暗号化器を設計する場合，通常，非常に大きな数を扱います．そのように大きな数をどのように処理するのかがポイントになります．ここでは，乗算器と大小比較器についてのみ述べましたが，ほんとうは，加算や減算についても，算術演算子をそのまま使用するのは好ましくありません．こ

図6　10進数の割り算

図7　12345÷171の結果

図8　2進数の割り算

のように見ていくと，多くの改善点があることが分かると思います．課題に取り組む際の参考にしてください．

5 RSA 暗号回路のインターフェース

RSA 暗号化器と復号器のインターフェースについて説明します．

暗号化器には，入力として，平文，暗号化鍵，法が，出力として，暗号文が，それぞれ必要になります．一方，復号器には，入力として，暗号文，復号鍵，法が，出力として，平文が，それぞれ必要になります．以下では，これらの信号線幅について述べます．

まず**図9**に，平文，暗号文，ブロック，文字の関係を示します．平文は通常，一度に暗号化するには，大きなサイズになっています．そのため平文を，一度に暗号化するサイズごとに区切ります．このときの一区切りを平文ブロックと呼ぶことにします．すなわち，一つ一つの平文ブロックが，式(1)におけるPとなります．なお同様に，暗号文を区切ったものを暗号文ブロックと呼ぶことにし，特に区別する必要がない場合は，単にブロックと呼ぶことにします．また各ブロックは，文字単位で区切ることができます．このとき，各ブロックに含まれる文字数がブロック・サイズです．以下では，ブロック・サイズをBと表すことにします．なお通常，暗号化器(復号器)への入力は，1文字単位とします．すなわち暗号化器(復号器)は，まず，Bクロックかけて，B文字を受け取ります．その後，受け取ったB文字を連接してブロックを構成してから，暗号化(復号)します．

次に，法Mを2進数で表したときのビット幅をW_Mと表すことにします．Mを法とするmod 演算のアルゴリズムの多くが，

$$2^{W_M-1} \leqq M < 2^{W_M}$$

が成り立つことを条件としたものであるので，以下でもこの条件が成り立つものとします．なお，この条件は，法MのMSB (Most Significant Bit；最上位ビット)が1になることを表しています．

すでに説明したように，式(1)，式(2)から，暗号化器と復号器は同じ構成になります．ここで，暗号化器を復号器としても使用できるようにするためには，入力される文字のビット幅と出力される文字のビット幅を一致させておく必要があります．そのビット幅をW_Cと表すことにします．ところで，式(1)，式(2)から，$P<M$，$C<M$が成り立つことが分かりますが，通常，暗号文ブロックのビット幅と法Mのビット幅は同じになります．このことは，先ほどの暗号化・復号の例を見れば分かりやすいと思います．先ほどの例の法$M=253$を2進数で表すと，11111101となり，8ビットになります．ここで例えば，2文字目「n」を暗号化した結果は，209となっています．この値はM未満ですが，2進数では11010001となり，8ビットになります．このことから，$W_C \times B = W_M$とすればよいことが分かります．先ほどの例は，$W_C=8$，$B=1$，$W_M=8$ということになります．なお先ほどの例の平文ブロックは，ASCIIコード1文字(7ビット)であり，8ビットに収まっています．

また，暗号化器を復号器としても使用できるようにするためには，さらに，暗号化鍵のビット幅と復号鍵のビット幅を比較して，長いほうのビット幅を採用する必要もあります．鍵の作り方から分かると思いますが，暗号化鍵Eと復号鍵Dの値は，法Mの値より小さくなるので，ここでは，これらの鍵のビット幅は，法Mのビット幅W_Mと同じにしておきます．ところで，暗号化器・復号器には，平文や暗号文を1文字単位で入力するのですから，暗号化鍵E，復号鍵D，法Mに関しても，W_CビットごとにB回に分けて入力した方がよいでしょう．

以上のように，実際の暗号化器・復号器では，平文ブロックP，暗号文ブロックC，暗号化鍵E，復号鍵D，法Mを，文字単位に分けて入出力することになります．そこで，文字単位に分けたものを，それぞれの記号の頭に「C」を付けて，CP，CC，CE，CD，CMと表すことにします．これ

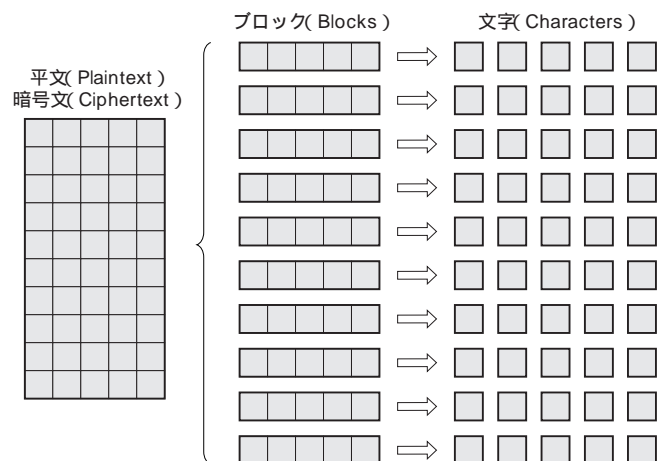



図9　平文，暗号文，ブロック，文字の関係

らを**表2**にまとめます．

以上のことから，RSA暗号化器・復号器のインターフェースは，**図10**のようになります．実際には，**図10**に示した信号線以外に，クロック信号やリセット信号，暗号化・復号の開始を表す信号，データの有効性を表すイネーブル信号などが必要です．また，実際の暗号化器・復号器では，処理するデータを受け取るのに要する時間と比べて，そのデータを暗号化・復号する時間の方がはるかに長くなります．そのため，暗号化器・復号器の内部にバッファを設け，バッファに空きがある間はデータを受信し，バッファが一杯のときは受信を拒否するなどの対策が必要になり，そのための制御信号なども追加されます．

6 課題

● Level 1：基本課題

基本課題では，タイミングに関する条件をほとんど指定しません．べき乗の計算およびmod演算の実現方法に注力してください．

基本課題のタイミング・チャートを**図11**に，ピン配置を**表3**に示します．なお，基本課題では，ブロック・サイズは固定で，$B=1$とします．すなわち，1文字ずつ暗号化・復号を行います．

図11に示すように，RSA暗号化器には，START信号と同時に，平文文字CP，暗号化鍵（1文字分）CE，法（1文字分）CMが入力されます．$B=1$なので，平文文字CP，暗号化鍵（1文字分）CE，法（1文字分）CMは，それぞれそのまま，平文ブロックP，暗号化鍵E，法Mになります．

RSA暗号化器は，これらの入力を受け取ってから，暗号化の処理を開始します．**図11**中の矢印は，暗号化を開始してから，結果を出力するまでのレイテンシ（遅延）を表しています．レイテンシは任意とします．

暗号化が終了すると，暗号文（1文字分）CCを出力します．それと同時に，その暗号文（1文字分）が有効なデータであること表すイネーブル信号CC_ENを出力します．なお，$B=1$なので，暗号文（1文字分）CCは，そのまま暗号文ブロックCになります．

表2　記号とビット幅

	記号	ビット幅	暗号化器	復号器
平文ブロック	P	$W_M(=W_C \times B)$	−	−
平文文字	CP	W_C	入力	出力
暗号文ブロック	C	$W_M(=W_C \times B)$	−	−
暗号文文字	CC	W_C	出力	入力
法	M	$W_M(=W_C \times B)$	−	−
法（1文字分）	CM	W_C	入力	入力
暗号化鍵	E	$W_M(=W_C \times B)$	−	−
暗号化鍵（1文字分）	CE	W_C	入力	−
復号鍵	D	$W_M(=W_C \times B)$	−	−
復号鍵（1文字分）	CD	W_C	−	入力
ブロック・サイズ	B	−	入力	入力

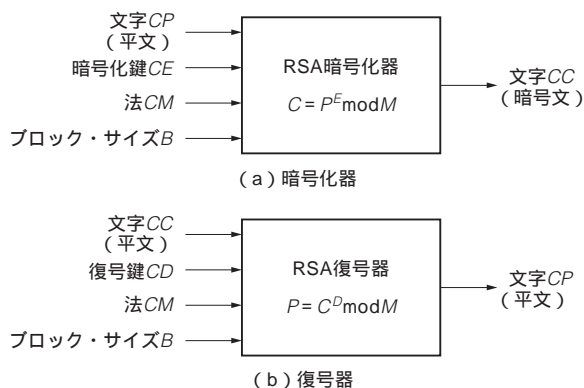
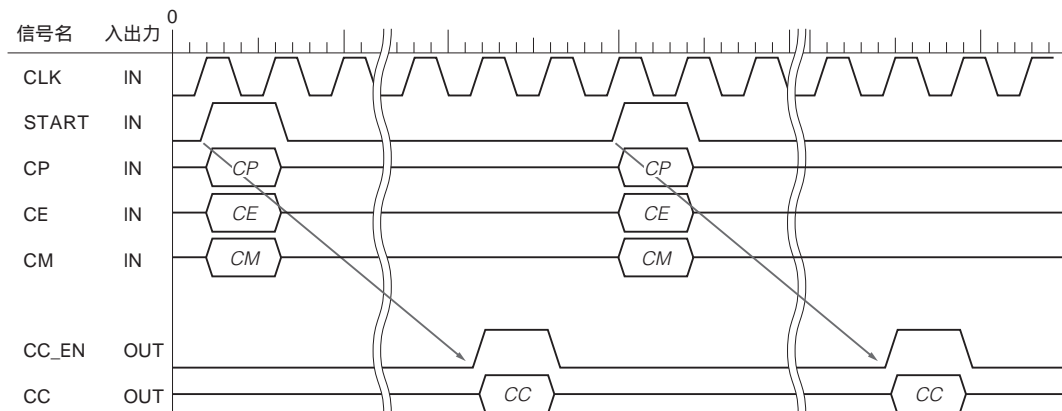



図10　RSA暗号化器・復号器のインターフェース

図11　基本課題のタイミング・チャート

暗号化処理中に，START信号が‘1’になることも考えられますが，その場合の対処も任意とします．途中までの処理を初期化して，新たに最初から暗号化処理を始めるようにしても構いませんし，暗号化処理が終わるまで，START信号を無視しても構いません．

暗号化結果および復号結果は，複数の鍵や法を使って確認してください．なお，新たに鍵や法を作成する場合，$128 \leqq M < 256$ となるようにしてください．暗号化結果および復号結果の確認方法も任意とします．例えば，テストベンチ内で，設計した回路を，暗号化器用と復号器用に二つインスタンス化し，暗号化結果が，そのまま復号器に入力されるようにしても構いません．その場合，もし必要であれば，暗号化器と復号器の間でやりとりする制御信号を追加することもOKです．もちろん，暗号化処理と復号処理を別々に確認しても構いません．

このほか，特に指定がない項目に関しては，基本的に任意とします．

● Level 2：応用課題

応用課題では，基本課題において固定値としていたブロック・サイズを可変とします．応用課題のタイミング・チャートを**図12**に，ピン配置を**表4**に示します．

図12および**表4**に示すように，応用課題では，ブロック・サイズ B（$1 \leqq B \leqq 4$）を2ビットの外部入力とします．ブロック・サイズ B も，START信号に同期しているものとします．このとき，B の値に応じて，B クロックかけて，B 文字の入力を受け取ります．最大4文字です．受け取ったすべての文字を，回路内部で連接してから，暗号化処理を開始します．この連接は，単純な連接（VHDLでは「&」，Verilog HDLでは「{}」）で構いません．もちろん，連接する前に処理を開始したり，単純な連接でなく，何らかの工夫を加えても構いません．また，文字の入力順序に関しても，上位文字から入力されることを想定しても，下位文字から入力されることを想定しても，どちらでもOKです．さらに，2ビットの2進数に対して，ブロック・サイズ B（$1 \leqq B \leqq 4$）の値をどのように割り当てるのかも自由です．

図12中の矢印は，暗号化を開始してから，結果を出力するまでのレイテンシ（遅延）を表しています．基本課題と同様に，レイテンシは任意とします．

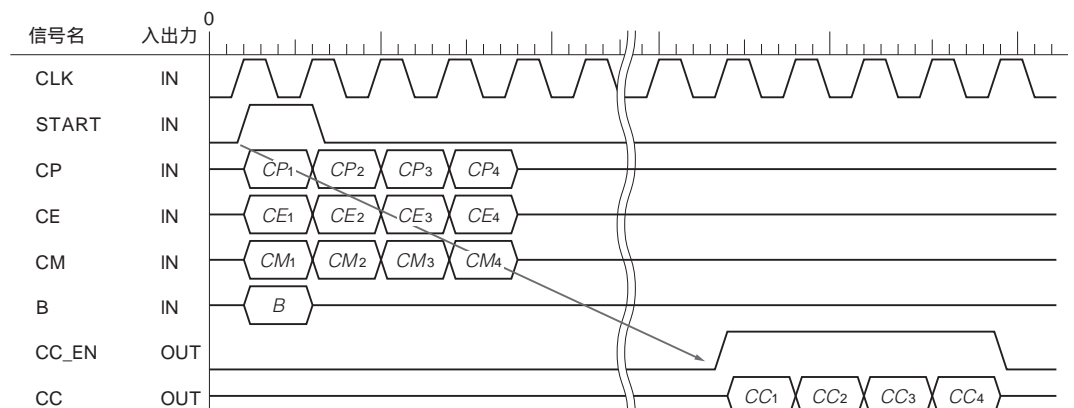
なお，基本課題同様，暗号化結果および復号結果は，複数の鍵や法を使って確認してください．**表5**に，ブロック・サ

表3　基本課題のピン配置

信号名	方　向	ビット幅	説　明
CLK	IN	1	クロック
RESET	IN	1	リセット
START	IN	1	暗号化開始（'1'で開始）
CP	IN	8	平文文字（1文字分）
CE	IN	8	暗号化鍵（1文字分）
CM	IN	8	法（1文字分）
CC_EN	OUT	1	暗号文出力中に'1'
CC	OUT	8	暗号文文字（1文字分）

表4　応用課題のピン配置

信号名	方　向	ビット幅	説　明
CLK	IN	1	クロック
RESET	IN	1	リセット
START	IN	1	暗号化開始（'1'で開始）
CP	IN	8	平文文字（1文字分）
CE	IN	8	暗号化鍵（1文字分）
CM	IN	8	法（1文字分）
B	IN	2	ブロック・サイズ（1～4の範囲で可変）
CC_EN	OUT	1	暗号文出力中に'1'
CC	OUT	8	暗号文文字（1文字分）

**図12
応用課題のタイミング・チャート**

イズ別の法Mの範囲および鍵の例を示しておきます．これ以外の鍵については，皆さん自身で作成してみてください．

このほか，信号の追加など，特に指定がない項目に関しては，基本的に任意とします．

● 社会人部門の条件

社会人部門は，課題をより実践的にするため，FPGAに実装することを前提として設計していただきます．

レポートには，FPGA設計ツールが出力したレポート・ファイル（使用論理ブロック数や使用専門機能ブロック数最大動作周波数が分かるもの）を添付してください．ターゲットFPGAの選択についても審査対象とします．設計意図，アーキテクチャ，コスト・パフォーマンスなどを考慮して，ターゲットFPGAを選択してください．

VHDL，Verilog HDL以外の設計言語による参加も認めます．

なお，2007年12月10日までの間に，FPGA評価ボードやFPGA開発ツールの貸し出しや応募時のレポート方法に関する情報の提供などを行う場合があります．本コンテストのWebサイトや本誌2007年12月号～2008年2月号に掲載する「コンテスト参加要領」を必ずご確認ください．

7 速度と規模の単位（学生部門対象）

筆者の勤務する大学では，論理合成用のツールとして米国Synopsys社の「Design Compiler」を使用していますが，このツールは誰もが使えるわけではありません．そこで，

- 2入力XORゲート一つの遅延時間を1UNIT_DELAY
- 2入力XORゲート一つの面積を1UNIT_AREA

とします．

具体的には，50入力XOR回路を合成していただき，その1段当たりの遅延時間を「単位時間」として，速度の単位とします．50入力XOR回路のVHDLソース・コードを**リスト1**に示します（このファイルも本コンテストのWebサイトからダウンロード可能）．

● 速度

Design Compilerでは，6段，49個のXOR回路が合成されます．クリティカル・パス遅延はreport_timingコマンドにより7.17であることが分かります．そこで7.17/6＝1.195を単位（1UNIT_DELAY）とします．例えば，ある遅延が20ならば，20/1.195＝17.74UNIT_DELAYということになります．

● 規模

面積はreport_areaコマンドのtotal cell areaにより147.0であることが分かります．XORゲート数は49個なので，147.0/49＝3.0を単位（1UNIT_AREA）とします．例えば，ある回路面積が200ならば，200/3.0＝66.67 UNIT_AREAとします．

● RAMやROMを用いる場合の注意

多めのデータを取り扱う回路では，メモリを使用する場合が想定されます．メモリを使う場合も，回路規模に含ま

表5 ブロック・サイズ別の鍵の例

ブロック・サイズB	法Mの範囲	鍵の例					
		p	q	M	L	E	D
1	$128 \leqq M < 256$	11	23	253	110	101	61
2	$32768 \leqq M < 65536$	127	433	54991	3024	73	1657
3	$8388608 \leqq M < 16777216$	2087	8017	16731479	8360688	31	3506095
4	$2147483648 \leqq M < 4294967296$	58771	70877	4165512167	2082691260	17	612556253

LSIデザインコンテスト・イン沖縄について

当初は琉球大学工学部の授業の一環として実施していたLSIデザインコンテスト・イン沖縄は，2008年で早いもので第11回を迎えます．その間に，国内の大学・高専からの参加が大幅に増えたばかりでなく，アジアを中心に韓国，インドネシア，ベトナムなどの海外の大学からの参加も順調に増え，今や100件以上もの応募がある大規模な設計コンテストと成長しました．

このように，国際的に成長したコンテストを祝して，2008年から優秀な設計チームに，電子情報通信学会やIEEEのアワードの贈呈を検討しています．こうした賞を目指し，学生の皆さんのこれまで以上にユニークな設計を期待しています．

れるように合成させてください．具体的には，メモリは合成可能なHDLコードで記述し，フリップフロップなどを用いる回路で実現してください．ROMテーブルを使用する場合も，マクロなどは使用せず，組み合わせ回路で実現してください．

ただし，FPGAのみをターゲットとする場合は，内蔵のメモリ・ブロックを使用しても構いません．その場合は，レポートに分かりやすく明示してください．

8 提出レポートについて

● 学生部門

今回から応募レポートは，IEEEの論文スタイルに統一します．http://www.ieee.org/portal/cms_docs_iportals/iportals/publications/journmag/transactions/TRANS-JOUR.docを参考にしてください．特に，ユニークな点や技術的に優れた点の説明を，十分かつ簡潔明瞭に記述してください．枚数は，最大で6ページまでとします．

海外からの審査員が理解できるように，できるだけ英語での執筆およびコンテスト当日の発表を勧めます．日本語でも構いませんが，国際化ということで多少の文法誤りは気にせずに，できるだけ英語にチャレンジしてみてください．

レポートはPDF化して，LSI-contest@dsp.cse.kyutech.ac.jpまで提出してください．締め切りは，2008年1月31日午後5時とします．

● 社会人部門

レポートは，ファイルによるE-mail送付または郵送で受け付けます．ファイルによる応募の場合は，レポートはPDFにしてください．

レポートには，FPGA設計ツールが出力したレポート・ファイルを添付するほか，設計した回路のレイテンシを必ず明記してください．書式については指定しませんが，日本語で，A4判用紙8ページ以内にまとめてください．レポートとは別に参考情報を添付いただくことは任意とします．

本コンテストは，必ずしも数値的な要素だけで優劣を決めるとは限りません．結果的に，提出していただくレポートそのものも評価対象となります．

締め切りは2008年1月31日（必着）です．

リスト1　50入力XOR回路のVHDLソース・コード

```
library  IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity PARITY is
    port ( A : in  unsigned(49 downto 0);
           Y : out std_logic );
end PARITY;

architecture RTL of PARITY is
begin

    process(A)
    variable TMP : std_logic;
    begin
        TMP := '0';
        for i in 0 to 49 loop
            TMP := TMP xor A(i);
        end loop;

      Y <= TMP;
    end process;
end RTL;
```

9 審査のポイント

速度や回路規模だけでなく，アーキテクチャのユニークさ，アイデアを十分に考慮して審査します．

社会人は，Design Wave設計コンテスト審査委員会が審査を行います．学生は，LSIデザインコンテスト実行委員会で1次審査を行い，上位10チーム程度を2008年3月に沖縄で開催予定のコンテスト発表会に招待します．そこで，プレゼンテーションによる最終審査を実施し，上位3チームに対しアワードを贈呈します．大学院生，学部学生，高専生のレベルに応じて審査します．

＊　＊　＊

本設計コンテストの学生部門は，主催：LSIデザインコンテスト実行委員会，共催：琉球大学情報工学科，沖縄産業振興センター，経済産業省九州経済局，九州半導体イノベーション協議会，電子情報通信学会スマートインフォメディア研究会，協賛：ソニーLSIデザイン，後援：CQ出版，国立沖縄工業高等専門学校，沖縄県，内閣府沖縄総合事務局により実施されています．

参考・引用＊文献

(1) 吉田たけお，尾知 博；VHDLで学ぶディジタル回路，CQ出版社．

よしだ・たけお
琉球大学工学部情報工学科
おち・ひろし
九州工業大学情報工学部電子情報工学科